VelociWatch: Designing and Evaluating a Virtual Keyboard for the Input of Challenging Text

Keith Vertanen, Dylan Gaines, Crystal Fletcher, Alex M. Stanage, Robbie Watling, Per Ola Kristensson

Department of Computer Science
Michigan Technological University
Houghton, Michigan, USA
{vertanen | dcgaines | tafletch | amstanag | rwatling}@mtu.edu

Department of Engineering
University of Cambridge
Cambridge, United Kingdom
pok21@cam.ac.uk

ABSTRACT

Virtual keyboard typing is typically aided by an auto-correct method that decodes a user's noisy taps into their intended text. This decoding process can reduce error rates and possibly increase entry rates by allowing users to type faster but less precisely. However, virtual keyboard decoders sometimes make mistakes that change a user's desired word into another. This is particularly problematic for challenging text such as proper names. We investigate whether users can guess words that are likely to cause auto-correct problems and whether users can adjust their behavior to assist the decoder. We conduct computational experiments to decide what predictions to offer in a virtual keyboard and design a smartwatch keyboard named VelociWatch. Novice users were able to use the features of VelociWatch to enter challenging text at 17 words-per-minute with a corrected error rate of 3%. Interestingly, they wrote slightly faster and just as accurately on a simpler keyboard with limited correction options. Our finding suggest users may be able to type difficult words on a smartwatch simply by tapping precisely without the use of auto-correct.

CCS CONCEPTS

• **Human-centered computing → Text input**.

KEYWORDS

Text entry; virtual keyboard; decoder; smartwatch

ACM Reference Format:
Keith Vertanen, Dylan Gaines, Crystal Fletcher, Alex M. Stanage, Robbie Watling, Per Ola Kristensson. 2019. VelociWatch: Designing and Evaluating a Virtual Keyboard for the Input of Challenging Text. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3290605.3300821

1 INTRODUCTION

Text entry is a ubiquitous computing activity. As interaction moves from desktop computers to mobile phones, smartwatches, and optical see-through head-mounted displays, the need for fast and accurate text entry remains. A particular challenge is the inherently noisier input that a text entry system has to deal with as text entry methods move beyond physical full-sized keyboards. Such physical keyboards accommodate ten-finger typing and provide tactile feedback. When such keyboards are transplanted to small touchscreens, the lack of tactile sensation feedback and small form factor result in increased noise as users' input becomes less precise.

A popular solution is to use an auto-correct algorithm to rectify typing mistakes. The algorithm's job is to infer the most likely intended text from a user's noisy input. When this method is probabilistic, it is called a *decoder*. The decoder searches for the most probable text hypothesis given an uncertain *observation sequence*. This is possible because natural languages are highly redundant with most letter sequences being improbable. Valid letter sequences can be captured by a statistical *language model*. The decoder's search for the most likely text is guided by this language model.

A well-designed decoder can help users reduce their error rate. This may also increase their entry rate by allowing faster and less precise typing. However, decoding can also result in unexpected results. When the decoder is carrying out its search, it is relying heavily on its language model. Thus if the user's intended text is similar to the text the language model has been trained on, the probability is high that the decoder will return the correct text. The flip-side is that if the user is writing text that is not well-represented in the language model, the decoder may return erroneous text. This results in the user being unexpectedly exposed to

Figure 1: Entering text on the VelociWatch keyboard (left) and on a simpler version of the keyboard (right).

an incorrect word, which has to be manually corrected. This "auto-correct trap" [31] increases error rates, reduces entry rates, and increases user frustration.

To investigate how to help users avoid and correct recognition errors, we created VelociWatch, a virtual keyboard optimized for the input of challenging text. We will compare the interface we designed (Figure 1 left) with a simpler version with more limited correction features (Figure 1 right).

Challenging Text Input

We specifically want to study interface designs that help users avoid or fluidly correct auto-correction errors. However, most text entry evaluations have users copy easy to remember text such as the MacKenzie phrase set [20] or the Enron mobile data set [28]. As we will show, these phrase sets are quite predictable under a well-trained language model. Combining a good language model with a high-performance decoder results in user evaluations in which participants rarely face recognition errors. We will first design a new phrase set containing text that is harder for a decoder to infer, thus serving as a more challenging evaluation task.

Aside from using challenging phrases, we also wanted to collect data with substantial input noise. Thus we conducted our investigation on a smartwatch. The small form factor makes precise typing difficult. Nevertheless, it is feasible to type on a full QWERTY virtual keyboard on a smartwatch if typing is aided by a decoder. Due to its reliance on decoding for effective text entry, we argue a smartwatch is a good testbed for investigating decoder and interface improvements. In the future we anticipate such harder cases of keyboard decoding will be even more relevant as users transition from mobile phones to wearable devices that rely on noisy sensing methods such as depth cameras.

Error Avoidance via Letter Locking

Many users today have substantial experience with touchscreen keyboards and their associated recognition errors. Even without intimate knowledge of how a decoder works, we conjectured users may often know a priori when an error is about to occur. To test this, we designed a simple error avoidance technique to see if users could anticipate problematic words and change their input behavior. Our *letter locking* method transfers control of the decoding process to the user. When a decoder considers a single observation (i.e. a touch point), it will explore different hypotheses for the observation (e.g. all letters adjacent to the key typed). When the user locks a letter, the decoder is prevented from performing this exploration and the hypothesis is fixed to the key touched. To assist users in locking the right letter, the nearest letter to a user's touch location is shown in a large font. The user locks the letter by maintaining contact with the touchscreen for an extended duration. After a time threshold has been exceeded, the preview changes color and the device vibrates to indicate the letter has been locked.

Letter locking allows a user to fluidly switch control between the decoder and the user. A user can take control of how each letter is input and thereby either override decoding altogether by locking every letter in the word, or steer the decoder by locking a subset of the letters in a word. Using our new phrase set, we show in Experiment 1 that users can anticipate difficult words and lock letters to significantly reduce their error rate at a modest reduction in entry rate.

Error Avoidance and Correction via Selection Slots

Virtual keyboards often display a number of selection slots above the QWERTY layout. Users tap a slot to take a certain action such as completing the word currently being typed or using the literal text nearest to each tap observation. While the use of selection slots is common in commercial keyboards and some research interfaces (e.g. [8]), we are not aware of any work comparing the efficacy of the many possible designs. It would be onerous to user test the many different options. Instead, in Experiment 2 we conduct simulations on data collected from previous studies on small form-factor virtual keyboards. By simulating optimal use of the slots, our work helps interface designers decide what slot-based error avoidance and correction features to include.

Tap versus Swipe Selection

As we will see, our offline experiments support including a rich set of error avoidance and correction options. Providing numerous error avoidance and correction features not only consumes valuable screen real estate, but also requires a way for users to select the different options. Ideally these features are selectable with little risk of making a selection error. This helps avoid a cascade of errors that can be extremely detrimental to text entry performance [13].

Virtual keyboards can support the input of entire words by a continuous swipe gesture [35]. They can also support other functionality via swipe such as using a right swipe to

recognize pending input [26, 30] or to control other keyboard functions [11]. Swipes have also been used to correct speech recognition errors [17, 22, 27]. We wondered if swipes could enhance selecting a small number of actions on a virtual keyboard. In Experiment 3, we explore swipe versus tap selection on a smartwatch keyboard we call VelociWatch.

Simple versus Complex Design

Our experiments led to the design of a reasonably complex keyboard squeezed onto a small touchscreen device. Having too many options could cause overheads that might actually reduce real-world performance. For example, users may spend too much time checking prediction slots. Also, users might simply prefer a design with fewer options or that consumes less screen space. In Experiment 4, we compare our best design with a simpler one with fewer features.

Contributions

(1) We create a new phrase set for exploring challenging text entry. Little work has investigated the input of difficult text. We are the first to do this combined with the challenging form factor of a smartwatch.
(2) In a study with 16 users, we show that users can predict words that will likely be problematic for a virtual keyboard decoder. Further, we show that users can adjust their input to help the decoder avoid errors.
(3) We compare 135 keyboard designs in computational experiments on 3 K recorded sentences. We show for the first time the impact of a literal slot (a common feature in commercial keyboards). We are not aware of any comparison of this scale on the role of prediction slots based on actual touchscreen typing data.
(4) In a study with 24 users, we compare tap and swipe selection of small touchscreen targets. While we thought swipe selection would be better, both performed similarly. This is a surprising and useful result; many smartwatch text entry interfaces have been designed around avoiding small buttons (e.g. [3, 10, 12, 23]).
(5) In a study with 14 users, we compare our VelociWatch keyboard against a simpler keyboard with only limited correction options. We found on the simpler keyboard users could modulate their typing to be accurate, relying on a literal slot to avoid the auto-correct trap. This is further evidence that small targets may be viable. The study also brings out interesting questions regarding cognitive overheads and correction behavior.

2 RELATED WORK

Mobile text entry methods have been extensively researched [19, 21, 37]. Today virtual keyboards dominate mobile touchscreen devices. Virtual keyboard typing can be improved using decoders. Goodman et al. [7] proposed a substitution-only decoder which corrects typing errors using a combination of a touch model and a language model. Kristensson and Zhai [16] proposed correcting keyboard typing using geometric pattern matching. Clawson et al. [4] presented a correction method for physical thumb keyboards based on key timings. An orthogonal approach to decoding keyboard typing is the gesture keyboard [15, 34, 36] in which users gesture through all the letters of a word on the keyboard.

Text entry and virtual keyboard typing have also been investigated for smartwatches (see [1] for a survey). Early interfaces focused on techniques to allow users to deterministically select the tiny keys on a smartwatch (e.g. ZoomBoard [23], Swipeboard [3], SplitBoard [12], and DualKey [10]). Other interfaces such as WrisText [6] and InclineType [9] allow users to enter text using wrist movements. The first study to demonstrate the viability of recognition-based typing on a smartwatch form-factor full QWERTY keyboard was the system VelociTap [30]. VelociTap used a decoder to infer users' intended text. Gordon et al. [8] later demonstrated WatchWriter, a smartwatch keyboard with decoder-supported tap input, gesture-keyboard input, and word prediction. WatchWriter uses two suggestion slots. One has the literal text typed by the user while the other contains the decoder's most likely prediction. Other smartwatch studies include Yi et al. [33] and Turner at al. [25].

Weir et al. [31] proposed improving decoding using two techniques. The first technique was to use Gaussian Process regression to create a more accurate touch model. The second technique was to allow users to continuously regulate their uncertainty using pressure. The harder a user pressed on a keyboard key, the less likely the decoder would change the key press into another key. This allowed users to regulate their uncertainty since pressing harder would concentrate the probability mass near the contact point while exerting low pressure would spread out the probability mass across many keys and thereby allow the decoder to search wider.

Arif and Stuerzlinger [2] used the time and/or movement signal in a touchscreen tap to infer if a user had tapped with additional pressure. They tested a keyboard that used a tap with additional pressure to bypass incorrect predictions. For the input of phrases with some out-of-dictionary words, this approach increased entry rate and decreased error rate.

In this paper we also allow users to regulate their uncertainty, but our design is different from prior work in two ways. First, letter locking is a discrete binary technique which either fixes a letter or leaves the decoder free to replace it. This means the effect of locking letters is easy to understand. Additionally, since individual letters can be locked, users can type quickly in the portion of a word that is likely easy to predict, while slowing down in more difficult areas (e.g. the "ii" in "desiigner"). Second, letter locking is accompanied by

clear visual and vibration interface feedback. This allows the user to be confident that a letter has indeed been locked. Due to these factors, we conjecture users will have an additional sense of agency in controlling the decoder.

Weir et al. [31] also investigated whether users could predict if a decoder would be able to correctly infer a phrase or not. They found that for phrases that were correctly inferred, users generally had a high ability to predict the outcome. However, for phrases that were incorrectly inferred, users tended to overestimate the ability of the decoder.

Text entry methods are typically evaluated using a transcription task in which participants copy memorable phrases as quickly and as accurately as possible. MacKenzie and Soukoreff [20] created a standard phrase set to improve the replicability of studies. Their phrase set consists of 500 phrases designed to be memorable, although this was never tested. Vertanen and Kristensson [28] later presented a phrase set verified to be memorable based on emails written by Enron employees on their BlackBerry mobile devices. A study later showed there was no significant difference in entry or error rates between these two phrase sets [14], although the latter has higher external validity.

In our studies, we required participants to memorize phrases as the smartwatch's screen size precluded showing phrases during text entry. Memorization results in somewhat faster entry rates at the cost of somewhat increased error rates [14]. As is common in text entry studies, we used memorization since we feel it is more similar to real-world input. A composition task would be even more realistic [29], but makes error rate harder to measure. Further, we worried participants might compose only easy-to-recognize text.

There has also been research on sampling memorable and representative phrases across languages [18, 24]. Yi et al. [32] proposed a word clarity metric based on geometric pattern matching [16] for sampling phrases containing words that might be confused due to their geometric proximity. In contrast, we present phrases designed to be easy or hard to recognize for a decoder that uses a statistical language model to guide its search. We will do this by selecting phrases based on whether they include out-of-vocabulary words which are harder for the decoder to recognize.

3 CHALLENGING PHRASE SET

We wanted phrases that would be challenging to recognize while still being memorable. We sourced our phrases from Twitter messages sampled during 2016. We parsed the tweets to find likely sentences based on capitalization and end-of-sentence punctuation. We generated a banned word list of 1,706 obscene words semi-automatically from a variety of sources. We removed sentences containing a banned word.

For text entry evaluations, typically we aim for memorable phrases to avoid participants needing to refer to the phrase during entry. Longer sentences are normally more difficult to memorize, so we removed sentences with more than 10 words. As in the MacKenzie phrase set [20], we focused on sentences with four or more words. Using a word list of 100 K English words, we created a set of 1.04 M sentences where all words were in-vocabulary. We created a second set of 141 K sentences that had a single out-of-vocabulary (OOV) word.

In-vocabulary phrases	Out-of-vocabulary phrases
I'm prettier than you.	Ready to meet Nanook!
Woke up still Loving y'all.	I voted for Brexit.
I want a Margarita.	Atletico ties it up!
This debate calls for vodka.	Is Rafa playing today?

Table 1: Examples from our challenging Twitter phrase set.

We fielded a random subset of 850 of these sentences to three to five Amazon Mechanical Turk workers. We only kept sentences that the majority of workers rated as easy to understand, judged as having no spelling errors, and that workers could type from memory with no errors (ignoring case and punctuation). These sentences were further manually reviewed by one of the authors to remove any remaining offensive sentences. The final phrase set had 213 OOV phrases (denoted TwitterOOV) and 194 in-vocabulary phrases (denoted TwitterIV)[1].

Table 1 shows some example phrases. The OOV phrases are the challenging part of the phrase set. We included the in-vocabulary set for comparative purposes. Phrases are in mixed case consisting of the letters A–Z plus apostrophe, comma, question mark, exclamation mark, and period.

One way to measure a text's difficult is the frequency of OOV words. Another way is to measure its *perplexity* under a language model. Perplexity measures the average number of possible next tokens (typically words or characters) given the previous tokens. Lower perplexity means the text is generally easier to recognize. We compared the perplexity of our new phrases against the MacKenzie phrase set [20] (denoted MacKenzie) and the `mem1-5` sets from the Enron mobile data set [28] (denoted EnronMem). For all sets, we removed sentences with numbers and stripped punctuation aside from apostrophe.

To measure average per-character perplexity, we used the character language model employed by our decoder (to be discussed shortly). As shown in Table 2, our TwitterOOV had the highest perplexity. Thus we anticipate it should be suitably challenging in comparison to other common phrase sets. We also computed the percentage of words that were OOV with respect to our 100 K word list. All phrase sets had nearly no OOV words except for the TwitterOOV set.

[1] http://keithv.com/data/twitter-phrases.zip

Phrase set	Perplexity	OOV rate	Words / phrase
TwitterIV	3.65	0.00%	6.69
TwitterOOV	5.34	14.71%	6.80
MacKenzie	4.59	0.07%	5.43
EnronMem	3.96	0.10%	5.31

Table 2: The per-character perplexity, percentage of words that were out-of-vocabulary (OOV), and words per phrase in different phrase sets.

We also note how EnronMem is more predictable on a per-character basis than MacKenzie. While in prior work [14] the choice of phrase set did not impact user performance in crowdsourced experiments on desktop keyboards, these results suggest language model-based input methods may also want to consider the predictability of phrases. Specifically, recognition-based input systems evaluated on EnronMem may appear more accurate in comparison to other systems evaluated on the more challenging MacKenzie phrases.

4 APPROACH

Armed with our Twitter phrase set, we set out to investigate interface interventions aimed at helping users avoid or correct recognition errors. Throughout this paper, we focus on input on a smartwatch since in past work small touchscreen keyboards have resulted in relatively high error rates even on easy Enron mobile phrases [26, 30]. This paper presents the following progression of four experiments:

- **Experiment 1** — We compare user and decoder performance both with and without a feature allowing users to lock in particular characters of their input.
- **Experiment 2** — We explore how to use a small number of suggestion slots to enable low error rate and high keystroke savings. We do this in computational experiments on thousands of sentences typed on small virtual touchscreen keyboards.
- **Experiment 3** — We use Experiments 1 and 2 to inform the design of our smartwatch virtual keyboard VelociWatch. We compare three different options for using tap and swipe actions to trigger interface actions.
- **Experiment 4** — We improve VelociWatch based on Experiment 3. We compare our keyboard design with a simpler design with only a few correction options.

5 EXPERIMENT 1: LETTER LOCKING

The goal of this study was to investigate if users could anticipate words likely to cause a recognition error and modulate their behavior to avoid them. Further, we wanted to see whether this additional user signal could help our decoder.

Keyboard design

We designed a virtual keyboard for the Android Sony SmartWatch 3. This watch has a screen area of 29 mm × 29 mm with a resolution of 320 × 320 pixels. The watch has a 4-core 1.2 Ghz ARM CPU with 512 MB of memory. Recognition occurs locally on the device.

The keyboard contains the letters A–Z plus apostrophe. There is no spacebar key. As shown in Figure 2, the keyboard occupies the lower portion of the screen and measures 29 mm × 13 mm. The keys are shown using white text on a grey background with no explicit visual key borders. This results in an effective key size of 2.9 mm × 4.3 mm.

The text result area above the keyboard shows previously recognized text as well as the nearest keys for the current unrecognized input sequence (Figure 2a). Swiping right causes the current tap sequence to be recognized (Figure 2b). During recognition, the screen background turns green and no input is accepted until recognition completes.

When a user's finger is in contact with the screen, the keyboard displays the nearest key in a large font over the text result area (Figure 2c). This allows the user to reposition their finger despite the visual occlusion caused by their finger. If a touch event lasts for 500 ms or longer, the letter popup turns orange and the watch vibrates (Figure 2d). This signals that whatever letter is under a user's finger when they lift up will be locked and not subject to auto-correction.

Swiping left deletes the previous tap from the observation sequence and deletes the nearest key label from the result text area. Swiping up moves to the next task. Swipes had to be 3 mm or longer to be recognized. Swipe direction was determined by the angle between a swipe's first and last point. Taps had to be within 3 mm of the top of the keyboard area to register. Swipes could occur anywhere on the screen.

In this study users could not delete or otherwise correct recognition results (e.g. via a backspace key or a word suggestion bar). We chose to do this to understand the performance of letter locking in isolation rather than how it might perform in combination with other error correction features.

Decoder

Our decoder is based on the noisy tap decoder VelociTap [30]. The original version of VelociTap inferred the most probable sentence given a sequence of two-dimensional tap locations. Here we use a modified version that recognizes input progressively, supporting input of a variable number of words (as in [5, 26]). For further details about the decoder's operation, see [26, 30]. Here we highlight the changes made to support letter locking and give details about the specific language models we used in this paper.

To support letter locking, each tap observation was augmented with a field indicating if a tap should be treated as

Figure 2: Example of writing "ready to meet nanook" in Experiment 1. (a) The keyboard has already recognized "ready to". The user tried to type "meet" but was somewhat inaccurate. (b) After swiping right, the nearest key text is replaced with the correct recognition result. (c) If a finger is down, the nearest key appears in a large font. (d) After a period of time, the letter changes to orange to signal it has been locked and no longer subject to auto-correction. (e) After locking the remaining letters, the user correctly obtains "nanook".

certain. This field was set by the virtual keyboard whenever a touch event lasted over a threshold time. One or more taps in a word could be locked. In the case of locked taps, we allowed the keyboard model to only generate the key closest to the touch up location. Other non-locked taps were probabilistically decoded as normal. So for example, a user might lock "n", "a", and "n" in "nanook" but quickly tap the final three letters. In this case, the decoder might return results such as "nanook", "nannie", "nannied", and "nancy", but would not be able to propose "rebook" or "cook".

We prevented the decoder from deleting locked observations. This ensured all locked letters had to appear in the recognition result. We also disallowed insertions between consecutive locked letters. This prevents the decoder from inserting guesses that might be probable under the language model, but which may contradict a user's intent. For example, a user might lock the "p" and "c" in "snapchat" hoping to get a single word. In this case we want to avoid the decoder being allowed to insert a space between the locked letters.

We trained our language models as in [26]. Our word model has 588 K *n*-grams and a gzipped ARPA text size of 5.5 MB. In Experiment 1, we used a character model with 766 K *n*-grams and a size of 6.3 MB. In Experiments 2 and 3, we used a slightly bigger character model with 963 K *n*-grams and a size of 9.0 MB. We found this bigger model improved recognition accuracy on the input from Experiment 1.

Study Procedure

We recruited 16 participants via convenience sampling. Participants completed an hour session and were paid $10. Participants were aged 18–27 (mean 19.1). 10 participants identified as male, 2 female, and the rest did not answer. 15 participants were right-handed. 10 participants had never used a smartwatch, 3 used one occasionally, and 3 used one frequently.

In Experiment 1, participants entered memorable phrases from our Twitter phrase set. Due to the small size of the smartwatch screen, we further limited the Twitter phrases to those with six or fewer words. We also removed phrases with acronyms as we worried users might incorrectly assume the lock feature was only for spelling out acronyms. In the end we obtained 43 OOV phrases and 81 in-vocabulary phrases.

Experiment 1 was a within-subject experiment with two counterbalanced conditions. In the NoLock condition, the feature allowing letters to be locked was disabled. Participants could still touch and hold to receive visual feedback of the key they were over, but the resulting touch up location was probabilistically decoded. In the Lock condition, participants could touch and hold to lock individual letters. We instructed participants they could lock any number of letters in a word, including all or none. The keyboard reminded participants on each phrase that they could lock letters.

Participants wore the watch on their non-dominant arm and rested their arm on a desk. At the start of each condition, participants practiced writing two OOV and two in-vocabulary phrases. Practice tasks throughout this paper were excluded from analysis. In each condition, participants received six OOV phrases and six in-vocabulary phrases. Participants saw a random subset of the phrase sets and never saw the same phrase more than once. The OOV and in-vocabulary phrases were mixed together at random.

Participants could spend as long as they wanted memorizing a phrase. The phrase disappeared and could not be referred to again once input began. After completing each transcription task, the participant was shown the reference text, the recognition text, their entry rate, and their error rate. If the error rate was above 5%, it was shown in red and the watch vibrated twice.

Results

Figure 3 displays the main results. Overall, participants used the lock feature. 11.8% of taps in Lock were *long taps* lasting over 500 ms compared to only 0.4% of taps in NoLock. This difference was statistically significant: dependent t-test, $t(15) = 8.41$, $r = 0.91$, $p < .001$.

We measured the *error rate* using Character Error Rate (CER). CER is the number of character insertions, deletions,

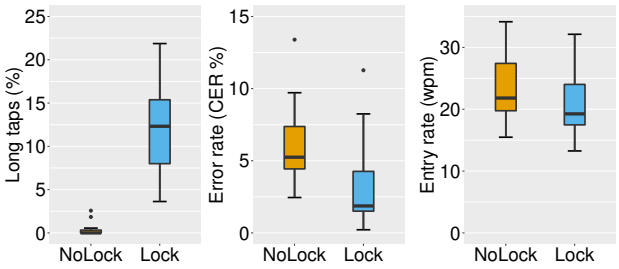

Figure 3: Long tap percentage, error rate, and entry rate with and without letter locking in Experiment 1.

and substitutions required to change the recognized text into the reference text, divided by the characters in the reference. Participants' use of letter locking reduced their error rate: 6.2% CER in NoLock versus 3.3% in Lock. This difference was significant: $t(15) = -4.21$, $r = 0.74$, $p < .001$.

We measured *entry rate* using words-per-minute (wpm) with a word being five characters including space. We calculated entry time from the first tap on the keyboard screen until the participant moved to the next phrase. As might be expected, locking letters reduced input speed: 20.9 wpm in Lock versus NoLock at 23.2 wpm. This difference was significant: $t(15) = -3.86$, $r = 0.71$, $p < .01$. Recognition time was negligible, 0.055 s in Lock and 0.063 s in NoLock.

Using left swipe to backspace tap events was infrequent in both conditions, 0.087 backspaces per character in the final text in NoLock and 0.081 in Lock. This difference was not significant: $t(15) = -0.53$, $r = 0.14$, $p = 0.604$. This indicates that despite often seeing visual feedback of incorrect nearest key text, participants tended to trust the auto-correction.

For tasks in Lock where the words in the phrase matched the number of recognition events, we determined which words (if any) a participant locked. We also determined if all the letters were locked or just a subset. Of the words with any locked letters, 35% had all their letters locked. The most frequent words that had all their letters locked were, in decreasing order: *bourre, brembo, auchinleck, haast, grigson, brofist, europe, delmon, evra, paak*. The most frequent in which any letter was locked were: *brembo, auchinleck, grigson, knapsacking, bourre, paak, europe, deano, luton, delmon*. With the exception of *europe* and *luton*, all these words were OOV. We conjecture participants mistakenly thought these two words needed locking to force recognition in lowercase.

We further analyzed participants' lock letter behavior in the Lock condition. We measured the percentage of in-vocabulary and out-of-vocabulary words where participants locked some letters (i.e. one or more) in a word and words where participants locked every single letter. Participants locked some letters in 4.8% of in-vocabulary words while they locked some letters in 75.6% of OOV words. This difference was significant: $t(15) = -9.79, r = 0.93, p < .001$. Participants locked all the letters in 2.1% of in-vocabulary words while they locked all the letters in 42.5% of OOV words. This difference was significant: $t(15) = -5.71, r = 0.83, p < .001$.

To lock a letter, participants had to precisely target a key. This alone could lead to a lower error rate. To measure if our decoder changes to support locking actually improved recognition accuracy, we ran offline experiments on the data from Lock. As in the user study, if the lock letter feature was turned on, the decoder locked the specific letters in a word where a user's tap exceeded the time threshold. If the lock letter feature was turned off, the decoder treated locked taps as normal probabilistic ones. We found turning the decoder feature off increased the error rate from 3.3% to 5.5%. Thus it was pinning portions of the observation sequence down that helped improve recognition accuracy.

Thus it appears without any explicit instructions, users could accurately predict which words might be problematic for an auto-correcting virtual keyboard. This could stem from previous exposure to auto-correction errors using virtual keyboards on mobile devices. It could also be that participants are singling out uncommon words based on their knowledge of English. An interesting observation is that participants often lock some but not all the letters in difficult words. We will analyze this in more detail in our final study.

Open comments were in general supportive of the lock feature such as "More effective with the ability to lock characters" and "Lock in is a vital feature for strange words and names." However some noted locking felt slow or suggested alternatives, e.g.: "I wish there was a way to lock a specific word after finishing it rather than locking each letter" and "I liked the lock feature. I just wish it was faster." In Experiment 3, we address these issues by lowering the time threshold and adding a way to select the literal text typed.

6 EXPERIMENT 2: SELECTION SLOT DESIGN

In this section, we conduct computational experiments to help design our keyboard. We explore four slot options for aiding users to avoid and correct recognition errors:

- **Prefix slots** — Word completions based on the currently noisy prefix input of a word. A user's taps thus far are treated probabilistically during the search for likely word completions.
- **Best slots** — Recognition alternatives based on all the pending taps. This is similar to a prefix slot other than the decoder is told to assume the taps represent an entire word and not necessarily a prefix.
- **Likely slots** — Whatever hypotheses have the highest probability regardless of whether they are prefix completions or recognition alternatives.
- **Literal slot** — The letters nearest to each tap.

Simulation and Test Data

We simulated a user who made optimal use of any available selection slots to try and obtain a sentence's reference text. For purposes of these experiments, we assume no use of other correction features such as backspacing and re-typing errors. We assume prefix predictions are made before even the first letter of a word is typed. Predictions use the previous selected words as context to the language model. If the simulated user is unable to get the desired reference word during input, we use the most likely hypothesis. This simulates leaving such errors uncorrected, potentially negatively impacting the language model's performance on future words.

We measure performance using two metrics. The first is the character error rate of the final text. The second is the potential *keystroke savings*. Keystroke savings (KS) is calculated as $KS = \left(1 - \frac{k_p}{k_a}\right) \times 100\%$ where k_p is the keystrokes required with word predictions and k_a is the keystrokes required without predictions. Higher keystroke savings is better. We assume slot selection requires one keystroke and adds any following space.

We ran our simulation on 3,158 sentences of virtual keyboard data from the Small and Tiny keyboard conditions of [30] (collected on a Nexus 4 phone) and all conditions of [26] (collected on a Sony SmartWatch 3). Note that some of these conditions involved users typing multiple words without explicitly denoting the space between words. We converted the data to word-at-a-time input by force-aligning the input sequences with the reference transcript.

Results

We simulated all possible ways to set a given number of slots to the prefix, best, likely, or literal options. The literal option was either included or not. If a candidate design had N slots of some type, we used the N most likely hypotheses of that type. We never included the same word in multiple slots so when filling a slot, we continued down the hypotheses for that type ordered by probability until we found a new word. Before a word's first tap, there are no best or literal options. In this case, we filled all slots with prefix completions.

Figure 4 shows the performance envelope of different slot designs. As expected, using more slots provides improved performance, but gains are marginal past five slots. The points in the lower left of Figure 4 represent designs where most or all slots are best slots. This resulted in poor keystroke savings but provided the lowest error rates for a given number of slots. Note that some keystroke savings was still possible due to prefix completions at the start of words.

For a given number of slots, the point with highest keystroke savings used most or all slots for prefix completions. The use of a literal slot in most cases resulted in worse performance in terms of error rate and keystroke savings. This is likely due to our test data consisting largely of Enron mobile phrases that had almost no OOV words.

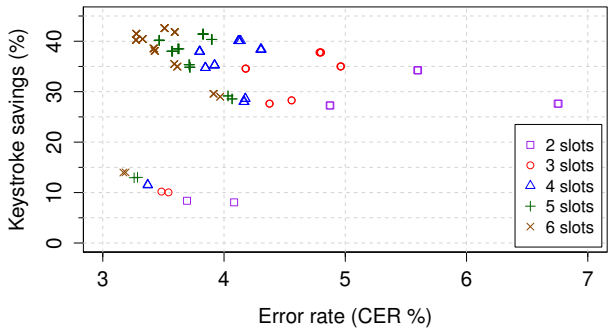

Figure 4: Performance of 2–6 slot virtual keyboard designs.

Number of slots				Keystroke	
Likely	Prefix	Best	Literal	savings	CER
0	0	2	0	8.4%	3.7%
0	0	1	1	8.1%	4.1%
0	1	0	1	27.6%	6.8%
1	0	0	1	27.6%	6.8%
0	0	4	1	12.9%	3.3%
2	2	1	0	40.2%	3.5%
3	**0**	**1**	**1**	**38.5%**	**3.6%**
1	4	0	0	41.4%	3.8%

Table 3: A selection of 2 and 5 slot designs. We selected the configuration in bold for our smartwatch keyboard.

Table 3 shows a selection of operating points for two and five slots keyboards (a complete list appears in our supplementary material). For our keyboard, we decided on five slots as it provided the majority of potential gains. Also a five slot design allowed inclusion of a literal slot with only a small performance penalty. This was important as we wanted to compare a literal slot against our lock letter method. Further, five slots allowed a visual layout in which a central option was surrounded by four other options. Out of the five slot designs, we decided on using a literal slot, three likely slots, and one best slot.

In Experiment 4, we will compare against a simpler two slot keyboard. With only two slots, using a literal slot requires choosing to either optimize keystroke savings or error rate. Using a likely or prefix slot with a literal slot provided nearly identical performance. For the two slot keyboard, we decided on using a likely slot as it allows word completions early in entry that hopefully converge to the best hypothesis as the entire word is typed. This choice of slots is similar to the WatchWriter [8] smartwatch keyboard which also used one literal and one likely slot.

7 EXPERIMENT 3: VELOCIWATCH

Based on Experiments 1 and 2, we designed a predictive virtual keyboard named VelociWatch. VelociWatch provides five text suggestions as well as a backspace action. In Experiment 3, we compare using swipes versus taps to select these options. We conjectured given the limited interaction size of a smartwatch, swipe selection would be more accurate.

Keyboard Design

We placed prediction slots in the four corners above and below the text area (Figure 1 left). This allowed an easy mapping between the visual location of the slots and swipe selection actions. Swiping diagonally up and left anywhere on the screen selected the top-left suggestion, swiping up and right selected the top-right, and so on. This led us to put the backspace key on the left side to correspond to left swiping to erase the last letter. The top-left slot was used for the literal slot. Similar to many commercial keyboards, we placed the literal text in double quotes. The most likely prediction was in the upper-right, the second most likely in the bottom-left, and the third most likely in the bottom-right.

We displayed the best recognition result in highlighted text in the text area. We worried placing a spacebar at the bottom of the keyboard would cause accuracy problems due to size limits and its proximity to the sensor edge and other letters. Spaces are common so having a sure way to type them is important, especially when other taps are treated probabilistically. We decided on a novel design in which the text area served as an implicit spacebar. This provided a large target away from the screen edge. It also felt natural to tap the highlighted best hypothesis in order to select it.

As in Experiment 1, dwelling in the keyboard area resulted in a large preview of the letter. Based on feedback from Experiment 1, we lowered the lock letter time threshold to 250 ms. The locked letters influenced the decoder's search and thus slots only showed suggestions complying with any locked letters. Before any input for a word, all four corner slots were populated with likely words given the previous text context. All slots updated every time the user typed a key. Recognition was performed on a separate thread so typing could proceed without waiting for recognition. Predictions were populated on average 91 ms after a key press.

Each prediction slot was 14 mm × 3.3 mm. The backspace button was 4.2 mm × 8.3 mm. The text area was 24 mm × 8.3 mm and could display three lines of text. After selecting a prediction, backspace, or the pending best text, the selected element would flash green to show which action was taken.

For this experiment, we tuned the parameters of VelociTap on 1,104 sentences of data collected on a Sony SmartWatch 3 in [26]. We used data from conditions in which participants had tapped one or two words prior to a recognition request.

Study Procedure

We recruited 24 participants via convenience sampling. None had participated in the previous study. Participants took part in a one-hour session and were paid $10. 22 participants were right-handed. Participants were aged 18–22 (mean 19.3). 15 participants identified as male, 6 female, and the rest did not answer. 13 participants had never used a smartwatch, 8 used one occasionally, and 3 used one frequently. 18 participants had not entered text on a smartwatch, 2 used speech recognition, and 4 used tap gestures on a QWERTY keyboard.

This was a within-subject experiment with three counterbalanced conditions. In the Tap condition, the best slot, the backspace, and the four likely/literal slots all had to be tapped. In the Swipe condition, the best slot was selected by right swipe, the backspace was selected by left swipe, and the four likely/literal slots were selected by diagonal swipes. In the Hybrid condition, the best slot was selected by right swipe, the backspace was selected by left swipe, and the four likely/literal slots were tapped.

Participants were first shown a short video demonstrating all the features of VelociWatch. This video showed both the swipe and tap actions for all features. At the start of each condition, participants were told whether swipe or tap was enabled for each feature. Participants wrote OOV and in-vocabulary Twitter phrases as in Experiment 1 except they wrote two practice phrases and 10 evaluation phrases. After each condition, participants completed a questionnaire.

Results

Figure 5 shows the main results. Overall, participants wrote at about the same speed in all three conditions: 17.7 wpm in Tap, 17.8 wpm in Swipe, and 17.4 wpm in Hybrid. This difference was not statistically significant: repeated measures ANOVA, $F_{2,46} = 0.26$, $p = 0.78$. The entry rate across all conditions for the in-vocabulary phrases was 21.3 wpm while for OOV phrases it was 13.9 wpm. Thus it appears users either slowed down their typing or had to perform corrective actions for the more difficult words.

We measured the error rate of the final text of each task after any user correction. Error rates were overall low: 3.2% CER in Tap, 3.4% in Swipe, and 2.1% in Hybrid. This difference was not significant: $F_{2,46} = 2.13$, $p = 0.13$. The error rate across all conditions for the in-vocabulary phrases was 2.0% CER while for OOV phrases it was 3.7%. This could mean that despite the error correction features, participants still could not obtain the specified text. Alternatively, harder phrases may have been more difficult to remember for participants.

We anticipated the small slot targets would result in more backspacing to correct errors resulting from errant selections. This did not seem to be the case, backspaces per character was similar in all conditions: 0.137 in Tap, 0.137 in Swipe,

and 0.151 in Hybrid. This difference was not significant: $F_{2,46} = 4.16$, $p = 0.66$. There was an increase in backspacing to 0.142 compared to 0.086 in Experiment 1 (averaged across conditions). Thus it appears participants were backspacing to some degree to correct erroneous slot selections.

Across all conditions, 3.3% of taps were over the lock letter threshold of 250 ms. This is a marked decrease from the 12% observed in the Lock condition of Experiment 1. This is likely due to competing ways to obtain the correct text via VelociWatch's slots. The lock letter feature appeared popular with some users but not others; half of users locked less than 1% of letters while the other half locked 1–14%.

Averaged over all conditions, we found participants selected the best slot 68% of the time, one of the likely slots 22% of the time, and the literal slot 10% of the time. For in-vocabulary phrases, the ratio of best, likely, and literal slot usage was 71%, 26%, and 3% respectively. But for OOV phrases, it was 65%, 19%, and 16%. While the literal slot was not that advantageous in Experiment 2, on more difficult text it became important for avoiding the auto-correct trap.

VelociWatch supports entering multiple words without spaces. 95.9% of slot selections were a single word, while 4.2% had multiple words. Just because the decoder inferred a multi-word result does not mean that was a user's intent; it could be a recognition error. We found 1.5% of the selections were a correct multi-word substring of the reference phrase. This indicates the decoder may have inserted spaces too freely. Further, only five participants selected correct multi-word results for more than 1% of their selections. These results suggest most participants preferred word-at-a-time entry.

When a likely slot was selected, we tallied how many taps a participant made prior to the selection: 0 taps 8.4%, 1 tap 6.5%, 2 taps 10.2%, 3 taps 14.6%, 4 taps 19.5%, 5 taps 11.6%, 6 taps 8.6%, 7 or more 20.6%. Thus occasionally making predictions before the start of the word was useful. Participants commonly typed several letters prior to selection.

On a 7-point Likert scale where 7 is strongly agree, the mean rating for the statement "I entered text *quickly*" was 5.2 in Tap, 5.4 in Swipe, and 5.6 in Hybrid. This difference was not significant, Friedman's test, $\chi^2(2) = 1.13$, $p = 0.57$. The mean rating for the statement "I entered text *accurately*" was 5.5 in Tap, 5.1 in Swipe, and 5.5 in Hybrid. This difference was not significant, Friedman's test, $\chi^2(2) = 1.19$, $p = 0.55$.

In open comments, some participants preferred swiping while other preferred tapping. A few participants commented they swiped when they were suppose to tap or vice versa. At the end of the study, participants specified their preferred method: 38% selected Swipe, 33% selected Hybrid, and 29% selected Tap. We asked participants how they dealt with difficult text. 15 participants commented specifically that they used the letter lock feature. Less common themes included checking the suggestions and using the literal text.

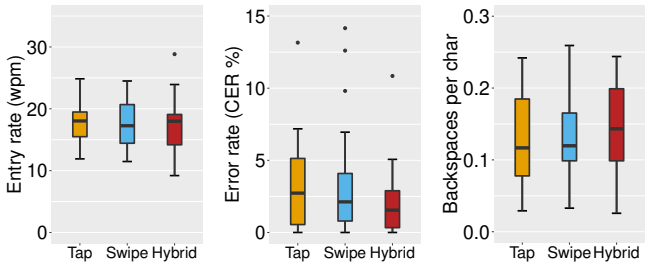

Figure 5: Entry rate, error rate, and backspaces per output character in Experiment 3.

8 EXPERIMENT 4: TWO SLOT KEYBOARD COMPARISON

In Experiment 3, there was no clear speed or accuracy advantage to swipe or tap selection. Given users were mixed in their preference, we decided to enable both swipe and tap functionality in Experiment 4. Further, we wanted to see how users would perform selections if given both options. In Experiment 3, we observed users frequently backspaced entire words. We added a feature allowing users to delete an entire word by long tapping on backspace. This resulted in a keyboard with a range of error correction features. In this section, we compare VelociWatch with a simpler keyboard that offers only a limited set of features.

Simpler Keyboard Design

The simpler keyboard has three options: the most likely word hypothesis on the left, the literal text in the middle, and a backspace button on the right (Figure 1 right). The most likely hypothesis could be either a prefix completion or a recognition alternative. We search for both types of hypotheses and use the one with highest probability. The backspace button deletes all pending letters for the current word. If there are no pending letters, it deletes the previous word. Note this design has no way to delete an individual character. Thus the keyboard could propose an incorrect prefix completion based on a user's completed noisy input of a word. In this case, the user's only recourse would be to start the word over.

We reduced the available input options and reduced visual clutter in a number of ways. We did not allow multi-word input. We also did not populate the slots until the first letter was typed. We turned off the lock letter feature and the key preview. The keyboard area was the same as VelociWatch. Prediction slots were 11.4 mm × 3.3 mm. The backspace key was 5.5 mm × 3.3 mm. We displayed up to three lines of text. Our design closely resembles the WatchWriter [8] interface except our keyboard was designed for a rectangular form factor and did not implement gesture-keyboard input.

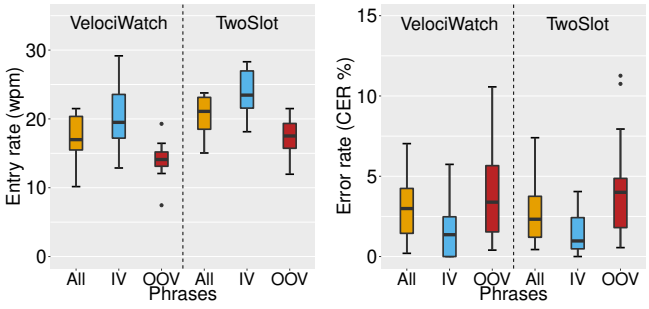

Figure 6: Entry and error rates in Experiment 4. Results over all phrases and the in-vocabulary (IV) and OOV phrases.

Condition	Entry rate (wpm)			Error rate (CER)		
	All	IV	OOV	All	IV	OOV
VelociWatch	17.3	20.6	14.0	3.0%	1.9%	4.2%
TwoSlot	20.6	24.1	17.1	3.0%	1.5%	4.4%
average	18.9	22.3	15.5	3.0%	1.7%	4.3%

Table 4: Entry and error rates in Experiment 4. Results over all phrases and the in-vocabulary (IV) and OOV phrases.

Some letters locked		All letters locked	
Input	Reference	Input	Reference
knovKAERT	knockaert	OZZY	ozzy
breXit	brexit	HAAST	haast
xgOPT	chopt	BROFIST	brofist
wEEzer	weezer	SEVCO	sevco
DESIIgner	desiigner	AUCKENLECH	auckenlech

Table 5: Example words where users locked some or all of the letters. Locked letters are shown in uppercase. Lowercase letters show the letter nearest to each non-locked tap.

Study Procedure

We recruited 14 participants via convenience sampling. None had participated in previous studies. Participants took part in a one-hour session and were paid $10. 13 participants were right-handed. Participants were aged 18–22 (mean 19.2). 10 participants identified as male, 3 female, and 1 did not answer. 6 participants had never used a smartwatch, 7 used one occasionally, and 1 used one frequently. 10 participants had not entered text on a smartwatch, 2 used speech recognition, and 1 used tap and swipe gestures on a QWERTY keyboard.

This was a within-subject experiment with two counterbalanced conditions: In the VelociWatch condition, participants used the VelociWatch keyboard. In the TwoSlot condition, participants used the two slot keyboard.

At the start of the study, participants watched a video demonstrating the keyboard assigned to their first condition. They then wrote four practice phrases using that keyboard. We then showed a video demonstrating the other keyboard and they wrote four practice phrases with that keyboard. Participants then wrote 10 OOV and 10 in-vocabulary phrases in each condition using a procedure similar to Experiment 3. After each condition, they completed a questionnaire.

Results

Figure 6 and Table 4 show our main results. Much to our surprise, participants wrote faster in TwoSlot at 20.6 wpm versus VelociWatch at 17.3 wpm. This difference was statistically significant: $t(13) = 3.78$, $r = 0.72$, $p < 0.01$. Averaged across conditions, entry rate was substantially faster for in-vocabulary phrases at 22.3 wpm versus 15.5 wpm for OOV phrases. VelociWatch was slower than TwoSlot for both in-vocabulary and OOV phrases. We conjecture overheads associated with monitoring the additional slots or locking letters contributed to the slower entry rates in VelociWatch.

Error rate was similar in both conditions with a CER of 3.0%. This difference was not significant: $t(13) = -0.11$, $r = 0.03$, $p = 0.91$. Averaged across conditions, CER was substantially higher for OOV phrases at 4.3% versus 1.7% for in-vocabulary phrases. VelociWatch and TwoSlot had similar error rates for the in-vocabulary and OOV phrases. It is surprising that despite TwoSlot having only a coarse-grained word deletion feature, participants achieved acceptable error rates on challenging text.

Both interfaces supported deleting all pending input or a previous word. This feature was used frequently at 1.4 times per phrase in TwoSlot, but was only used 0.03 times per phrase in VelociWatch. It seems participants either preferred to backspace one letter-at-a-time, or did not learn this feature. Only VelociWatch allowed backspacing of individual letters. Participants used 0.12 backspaces per output character, similar to what we observed in Experiment 3.

In VelociWatch, participants used long taps to lock letters 3.0% of the time, similar to Experiment 3. We analyzed all the individual recognition events that had one or more taps in VelociWatch. 4.8% of words had at least one letter locked. 1.5% of words had all their letters locked. Table 5 shows some samples of each case.

Participants selected the best slot 46.4% of the time, one of the likely slots 38.4% of the time, and the literal slot 15.2% of the time. In TwoSlot, participants used the likely slot 61.9% of the time and the literal slot 38.1% of the time. It seems without other correction features, participants in TwoSlot often relied on careful typing and using the literal slot.

In VelociWatch, 60.3% of selections used tap while 39.7% used swipe. For tap selections, 37.8% were for a likely or literal slot, 27.7% for the best slot, and 34.5% for backspace (character or word). For swipe selections, 10.0% were for a likely or literal slot, 6.2% for space, and 83.7% for backspace. Thus when allowed to either tap or swipe, participants tended to use tap, but swipe use was still frequent especially for erasing previous characters. This suggests a virtual keyboard designer may want to consider adding swipe actions for some operations.

In VelociWatch, 91.9% of selections had one word while 8.1% had multiple words. The majority of multi-word recognitions matched the reference, 5.7% of total selections. This increase from Experiment 3 was driven by three participants who used multi-word input for 10%, 22%, and 40% of inputs.

On a 7-point Likert scale where 7 is strongly agree, the mean rating for the statement "I entered text *quickly*" was 5.1 in VelociWatch and 5.4 in TwoSlot. This difference was not significant, Friedman's test, $\chi^2(1) = 0.82$, $p = 0.37$. The mean rating for the statement "I entered text *accurately*" was 4.8 in VelociWatch and 4.6 in TwoSlot. This difference was not significant, Friedman's test, $\chi^2(1) = 0.00$, $p = 1.00$.

Eight participants preferred VelociWatch while six preferred TwoSlot. Participants preferring VelociWatch cited reasons such as lock letter, swipe, more suggestions, and character backspace. Participants preferring TwoSlot said it was simpler, had fewer options to scan, and resembled familiar keyboards. Common strategies mentioned for entering difficult words included typing slowly and locking letters.

9 DISCUSSION AND LIMITATIONS

Our most interesting finding was that the simple two slot keyboard out-performed our more feature-rich keyboard that we painstakingly designed via computational and user experiments. It did this for the input of difficult text in which half the phrases involved OOV words. This is quite remarkable since these OOV words were unlikely to be correctly predicted by the decoder. This left the user with only one choice: tap every letter of a word correctly with zero mistakes on a smartwatch. Due to the keyboard only having a word-level backspace, any errant tap required starting the entire word over. On just the OOV phrases, participants achieved an acceptable error level of 4.4% while still typing at 17.1 wpm.

When asked how they handled difficult words, 8 of the 14 participants mentioned they slowed down. This is corroborated by an analysis of the logs files. When entering in-vocabulary words using TwoSlot, participants wrote at 23.0 wpm. When entering OOV words, they slowed to 14.5 wpm. For OOV words, participants used the literal slot 84.7% of the time. For in-vocabulary words, they only used the literal slot 33.2% of the time. It appears participants could often enter OOV words simply by carefully tapping each letter and selecting the literal slot. It appears that even on a small virtual keyboard, users were able to exactly target all the letters of many words without the help of an auto-correct algorithm. This suggests precise target acquisition on capacitive touchscreens may not be that big of a problem.

As shown in Table 4, entry rates slowed substantially for phrases with an OOV word. Supporting faster entry for OOVs is challenging since users often slow down to be more accurate, a motor control reality. Reducing overheads associated with letter locking could speed entry, e.g. using a lower time threshold or using an input signal that is not time dependent such as force. Avoiding OOVs in the first place by expanding the vocabulary could also be advantageous. However, this requires care as we only want to add words that are new proper names or slang while avoiding adding words that are simply typos. Adapting the language model could also help here, allowing more approximate input of OOVs once a user has typed an OOV several times.

Our participants consisted of university-age students with substantial touchscreen typing experience. The mean rating for the statement "I frequently enter text on a mobile phone" was 6.7 (where 7=strongly agree). Despite half our participants having never used a smartwatch, they seemed to quickly learn to accurately target keys on its small touchscreen. Participants also seemed quite capable of anticipating auto-correction problems. This is likely a result of substantial first-hand experience. It remains to be seen if such precise typing or error awareness holds for a broader population or in more challenging scenarios (e.g. typing while walking).

Our studies all consisted of a single one-hour session. It is possible VelociWatch would become faster with practice. However, it is also possible inherent cognitive overheads related to monitoring more suggestion slots or deciding between corrective options slowed performance. While our design offered a theoretical keystroke savings of 39%, this assumes users always notice correct predictions. These aspects were not modeled as part of the computational-driven design of our interface. It would be interesting, but challenging, to incorporate such aspects into the simulations.

In Experiment 4 using VelociWatch, users locked only 3% of taps and the delete word feature was rarely used (0.03 times per phrase). Users preferred to delete characters instead. The simpler keyboard could delete only entire words. It could be that by forcing users to correct by deleting an entire word and retyping actually was faster in comparison to backspacing character-by-character. This suggests the delete word feature may need to be more prominent and faster to trigger.

We designed our keyboard for entering challenging text on a small touchscreen. Our choices may not be optimal for larger devices or for the input of more predictable text. There is scope for designs between our two keyboard interfaces.

Participants indicated they would have liked some of VelociWatch's features in the simpler two slot keyboard. This included backspacing individual letters, the option to swipe, and the ability to lock letters. Adding a targeted set of such features could result in a keyboard that performs better on challenging text than either of the two designs we tested.

10 CONCLUSIONS

We presented a new phrase set of memorable Twitter messages that provides challenging text for evaluating text entry interfaces. We investigated a new letter locking feature that allows users to take control of the recognition process. In our first user study, we found that users had a good sense of what words would be difficult for the auto-correct algorithm. Letter locking provided improved recognition accuracy with modest reductions in entry rate. We used data from thousands of sentences typed on small virtual keyboards to explore what suggestions to offer on a virtual keyboard. We found using a literal slot, the best recognition hypothesis, and three likely alternatives offered the best potential of fast and accurate input.

Combining our letter locking feature and our selection slot findings, we designed a smartwatch keyboard. We investigated using swipes rather than taps for selecting the keyboard's suggestions. We also tested using the text area as a large implicit spacebar. In our second user study, participants performed similarly using swiping or tapping. User opinion was mixed with some preferring swiping and some preferring tapping.

In our final user study, we compared our VelociWatch keyboard against a simpler keyboard with only two suggestion slots. Empirically, the simpler keyboard was faster at 20.6 wpm compared to our keyboard at 17.3 wpm. Both keyboards achieved a low corrected error rate of 3.0%. We think this is impressive performance given the challenging nature of our phrases and the noisy input resulting from a small touchscreen. Despite the somewhat slower entry rate, participants perceived VelociWatch to be of similar speed and accuracy. The majority subjectively preferred VelociWatch. The performance of the simpler keyboard suggests accurate typing for occasional difficult words may be possible even without auto-correction.

11 ACKNOWLEDGMENTS

This work was supported by Google Faculty awards (K.V. and P.O.K.), and EPSRC grants EP/N010558/1, EP/N014278/1, EP/R004471/1 (P.O.K.). The supplementary material for this paper contains participant data from Experiments 1–4, the instructional videos shown in Experiments 3 and 4, and the challenging Twitter phrase set.

REFERENCES

[1] Ahmed Sabbir Arif and Ali Mazalek. 2016. A Survey of Text Entry Techniques for Smartwatches. In *Proceedings, Part II, of the 18th International Conference on Human-Computer Interaction. Interaction Platforms and Techniques - Volume 9732*. Springer-Verlag New York, Inc., New York, NY, USA, 255–267. https://doi.org/10.1007/978-3-319-39516-6_24

[2] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2013. Pseudo-pressure Detection and Its Use in Predictive Text Entry on Touchscreens. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration (OzCHI '13)*. ACM, New York, NY, USA, 383–392. https://doi.org/10.1145/2541016.2541024

[3] Xiang 'Anthony' Chen, Tovi Grossman, and George Fitzmaurice. 2014. Swipeboard: A Text Entry Technique for Ultra-small Interfaces That Supports Novice to Expert Transitions. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 615–620. https://doi.org/10.1145/2642918.2647354

[4] James Clawson, Kent Lyons, Alex Rudnick, Robert A. Iannucci, Jr., and Thad Starner. 2008. Automatic Whiteout++: Correcting mini-QWERTY Typing Errors Using Keypress Timing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 573–582. https://doi.org/10.1145/1357054.1357147

[5] John J. Dudley, Keith Vertanen, and Per Ola Kristensson. 2018. Fast and Precise Touch-Based Text Entry for Head-Mounted Augmented Reality with Variable Occlusion. *ACM Transactions on Computer-Human Interaction (TOCHI)* 25, 6, Article 30 (12 2018), 40 pages. https://doi.org/10.1145/3232163

[6] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang 'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. WrisText: One-handed Text Entry on Smartwatch Using Wrist Gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, 181:1–181:14. https://doi.org/10.1145/3173574.3173755

[7] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language Modeling for Soft Keyboards. In *Eighteenth National Conference on Artificial Intelligence (AAAI '02)*. American Association for Artificial Intelligence, Menlo Park, CA, USA, 419–424. http://dl.acm.org/citation.cfm?id=777092.777159

[8] Mitchell Gordon, Tom Ouyang, and Shumin Zhai. 2016. WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3817–3821. https://doi.org/10.1145/2858036.2858242

[9] Timo Götzelmann and Pere-Pau Vázquez. 2015. InclineType: An Accelerometer-based Typing Approach for Smartwatches. In *Proceedings of the XVI International Conference on Human Computer Interaction (Interacción '15)*. ACM, New York, NY, USA, Article 59, 4 pages. https://doi.org/10.1145/2829875.2829929

[10] Aakar Gupta and Ravin Balakrishnan. 2016. DualKey: Miniature Screen Text Entry via Finger Identification. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 59–70. https://doi.org/10.1145/2858036.2858052

[11] Minako Hashimoto and Masatomo Togasi. 1995. A Virtual Oval Keyboard and a Vector Input Method for Pen-based Character Input. In *Conference Companion on Human Factors in Computing Systems (CHI '95)*. ACM, New York, NY, USA, 254–255. https://doi.org/10.1145/223355.223661

[12] Jonggi Hong, Seongkook Heo, Poika Isokoski, and Geehyuk Lee. 2015. SplitBoard: A Simple Split Soft Keyboard for Wristwatch-sized Touch Screens. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1233–1236. https://doi.org/10.1145/2702123.2702273

[13] Clare-Marie Karat, Christine Halverson, Daniel Horn, and John Karat. 1999. Patterns of Entry and Correction in Large Vocabulary Continuous Speech Recognition Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 568–575. https://doi.org/10.1145/302979.303160

[14] Per Ola Kristensson and Keith Vertanen. 2012. Performance Comparisons of Phrase Sets and Presentation Styles for Text Entry Evaluations. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces (IUI '12)*. ACM, New York, NY, USA, 29–32. https://doi.org/10.1145/2166966.2166972

[15] Per Ola Kristensson and Shumin Zhai. 2004. SHARK^2: A Large Vocabulary Shorthand Writing System for Pen-based Computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST '04)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/1029632.1029640

[16] Per Ola Kristensson and Shumin Zhai. 2005. Relaxing Stylus Typing Precision by Geometric Pattern Matching. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI '05)*. ACM, New York, NY, USA, 151–158. https://doi.org/10.1145/1040830.1040867

[17] Kazutaka Kurihara, Masataka Goto, Jun Ogata, and Takeo Igarashi. 2006. Speech Pen: Predictive Handwriting Based on Ambient Multimodal Recognition. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems (CHI '06)*. ACM, New York, NY, USA, 851–860. https://doi.org/10.1145/1124772.1124897

[18] Luis A. Leiva and Germán Sanchis-Trilles. 2014. Representatively Memorable: Sampling the Right Phrase Set to Get the Text Entry Experiment Right. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1709–1712. https://doi.org/10.1145/2556288.2557024

[19] I Scott MacKenzie and R William Soukoreff. 2002. Text entry for mobile computing: Models and methods, theory and practice. *Human–Computer Interaction* 17, 2-3 (2002), 147–198. https://doi.org/10.1080/07370024.2002.9667313

[20] I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase Sets for Evaluating Text Entry Techniques. In *Extended Abstracts on Human Factors in Computing Systems (CHI EA '03)*. ACM, New York, NY, USA, 754–755. https://doi.org/10.1145/765891.765971

[21] I. Scott MacKenzie and Kumiko Tanaka-Ishii. 2007. *Text Entry Systems: Mobility, Accessibility, Universality*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[22] J. Ogata and M. Goto. 2005. Speech Repair: Quick Error Correction Just by Using Selection Operation for Speech Input Interfaces. In *Proceedings of the International Conference on Spoken Language Processing*. 133–136.

[23] Stephen Oney, Chris Harrison, Amy Ogan, and Jason Wiese. 2013. ZoomBoard: A Diminutive Qwerty Soft Keyboard Using Iterative Zooming for Ultra-small Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2799–2802. https://doi.org/10.1145/2470654.2481387

[24] Germán Sanchis-Trilles and Luis A Leiva. 2014. A systematic comparison of 3 phrase sampling methods for text entry experiments in 10 languages. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services (MobileHCI '14)*. ACM, New York, NY, USA, 537–542. https://doi.org/10.1145/2628363.2634229

[25] Colton J. Turner, Barbara S. Chaparro, and Jibo He. 2017. Text Input on a Smartwatch QWERTY Keyboard: Tap vs. Trace. *International Journal of Human-Computer Interaction* 33, 2 (2017), 143–150. https://doi.org/10.1080/10447318.2016.1223265 arXiv:http://dx.doi.org/10.1080/10447318.2016.1223265

[26] Keith Vertanen, Crystal Fletcher, Dylan Gaines, Jacob Gould, and Per Ola Kristensson. 2018. The Impact of Word, Multiple Word, and Sentence Input on Virtual Keyboard Decoding Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 626, 12 pages. https://doi.org/10.1145/3173574.3174200

[27] Keith Vertanen and Per Ola Kristensson. 2009. Parakeet: A Continuous Speech Recognition System for Mobile Touch-screen Devices. In *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI '09)*. ACM, New York, NY, USA, 237–246. https://doi.org/10.1145/1502650.1502685

[28] Keith Vertanen and Per Ola Kristensson. 2011. A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices & Services (MobileHCI '11)*. ACM, New York, NY, USA, 295–298. https://doi.org/10.1145/2037373.2037418

[29] Keith Vertanen and Per Ola Kristensson. 2014. Complementing Text Entry Evaluations with a Composition Task. *ACM Transactions of Computer Human Interaction* 21, 2, Article 8 (February 2014), 33 pages. https://doi.org/10.1145/2555691

[30] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating Fast Mobile Text Entry Using Sentence-Based Decoding of Touchscreen Keyboard Input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 659–668. https://doi.org/10.1145/2702123.2702135

[31] Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. Uncertain Text Entry on Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2307–2316. https://doi.org/10.1145/2556288.2557412

[32] Xin Yi, Chun Yu, Weinan Shi, Xiaojun Bi, and Yuanchun Shi. 2017. Word Clarity As a Metric in Sampling Keyboard Test Sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4216–4228. https://doi.org/10.1145/3025453.3025701

[33] Xin Yi, Chun Yu, Weinan Shi, and Yuanchun Shi. 2017. Is it too small?: Investigating the performances and preferences of users when typing on tiny QWERTY keyboards. *International Journal of Human-Computer Studies* 106, Supplement C (2017), 44 – 62. https://doi.org/10.1016/j.ijhcs.2017.05.001

[34] Shumin Zhai and Per-Ola Kristensson. 2003. Shorthand Writing on Stylus Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, New York, NY, USA, 97–104. https://doi.org/10.1145/642611.642630

[35] Shumin Zhai and Per Ola Kristensson. 2012. The word-gesture keyboard: reimagining keyboard interaction. *Commun. ACM* 55, 9 (2012), 91–101.

[36] Shumin Zhai and Per Ola Kristensson. 2012. The Word-gesture Keyboard: Reimagining Keyboard Interaction. *Commun. ACM* 55, 9 (September 2012), 91–101. https://doi.org/10.1145/2330667.2330689

[37] Shumin Zhai, Per-Ola Kristensson, and Barton A Smith. 2005. In search of effective text input interfaces for off the desktop computing. *Interacting with computers* 17, 3 (2005), 229–250. https://doi.org/10.1016/j.intcom.2003.12.007